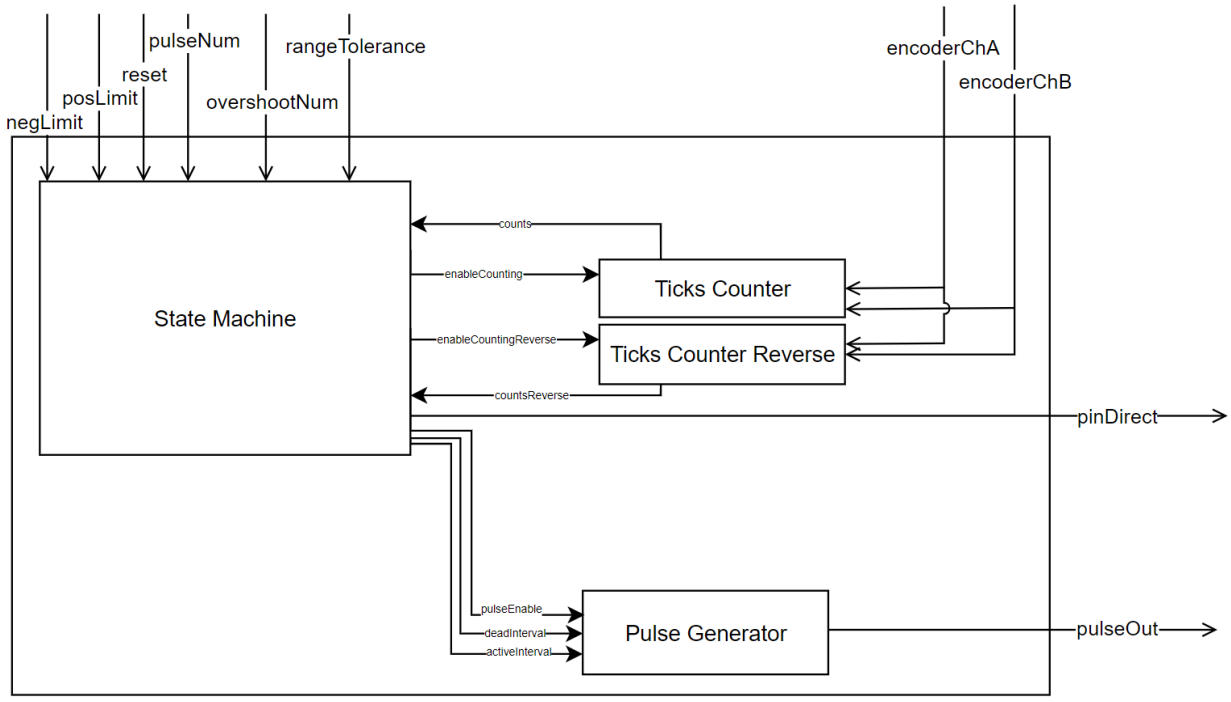
DLO MCC Dev Documentation

1. MCC Block Diagram

The diagram provided illustrates the essential components of the MCC block, consisting of four main parts: the state machine (controller), two tick counters for tracking encoder pulse counts, and a pulse generator responsible for actuator movement.

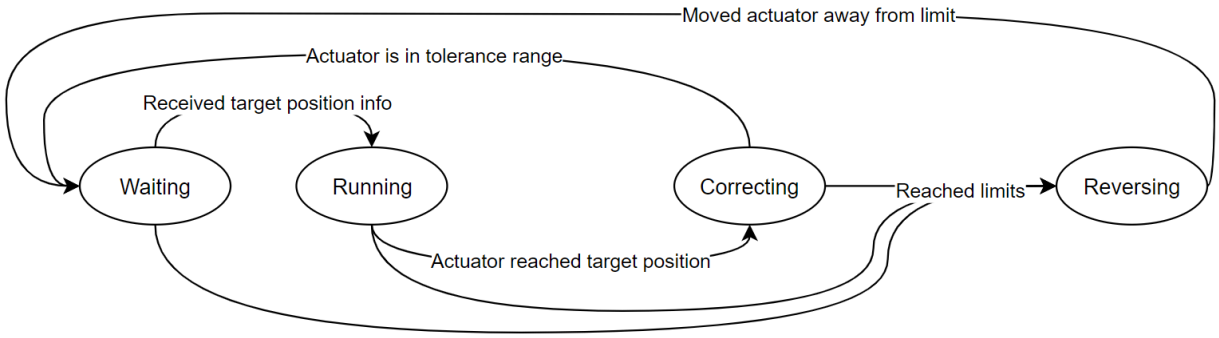
Overall, this block features 8 inputs and 2 outputs. While three inputs might seem complex, here are clarifications: **`pulseNum`** signifies the desired distance for the actuator's movement, **`overshootNum`** represents the intentional excess movement during actuation, and **`rangeTolerance`** establishes a narrow range around the target position. When the encoder pulse count falls within this tolerance range, the correction process concludes.



Port Names	Port Connections	Explanations
clk	clk	Clock input
reset	Control0(0)	The Control0's Least Significant Bit (LSB) is utilized for MCC reset functionality
deadIntervalRun	Control1(15 downto 0)	Adjust the dead width and pulse width of the pulse train signal while in the Running state
activeIntervalRun	Control1(31 downto 16)	
deadIntervalCorrect	Control2(15 downto 0)	Adjust the dead width and pulse width of the pulse train signal while in the Correcting state
activeIntervalCorrect	Control2(31 downto 16)	
pulseNum	signed(Control3(31 downto 0))	Desired number of encoder pulses for the target position
overshootNum	signed(Control4(31 downto 0))	Additional encoder pulses number for intentional overshooting
backPulseNum	Control5(31 downto 0)	Shifting the actuator away from the limits by a specific count of encoder pulses
rangeTolerance	Control6(31 downto 0)	The tolerance window size around the target position (minimum: 1)
waitInterval	Control7(31 downto 0)	Pause the system's operation. Wait for the encoder channels stabilize
encoderChA	InputA(0)	Encoder channel A
encoderChB	InputA(1)	Encoder channel B
posLimit	InputA(2)	Positive limit switch
negLimit	InputA(3)	Negative limit switch
pinDirect	OutputA(4)	Direction control pin
pulseOut	OutputA(5)	Pulse train output
stateOut(0)	OutputA(10)	State output - lower bit
stateOut(1)	OutputA(11)	State output - higher bit

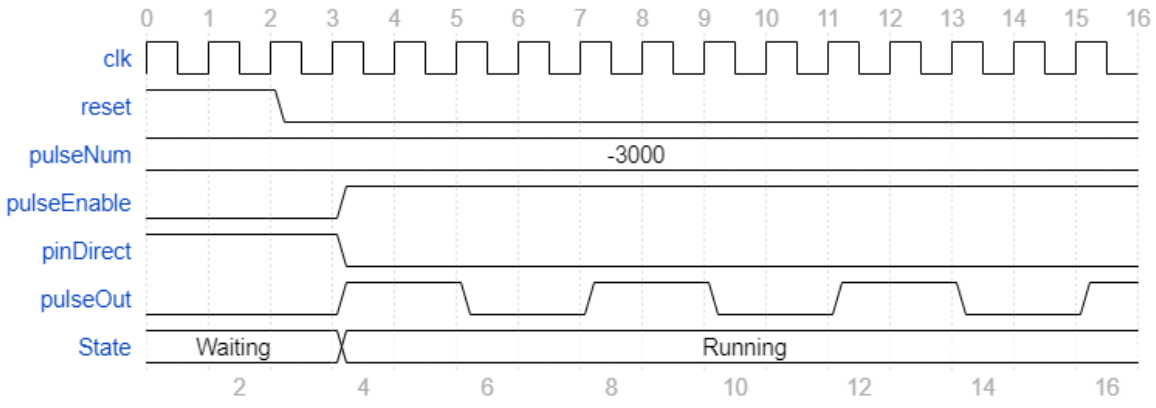
2. State Machine

This MCC operates as a four-state state machine, including Waiting, Running, Correcting, and Reversing states. In this context, the term "target position" specifically refers to the entered encoder pulse edge counts. The MCC block doesn't record the pulse output; instead, it tracks the quantity of received encoder pulse edges.

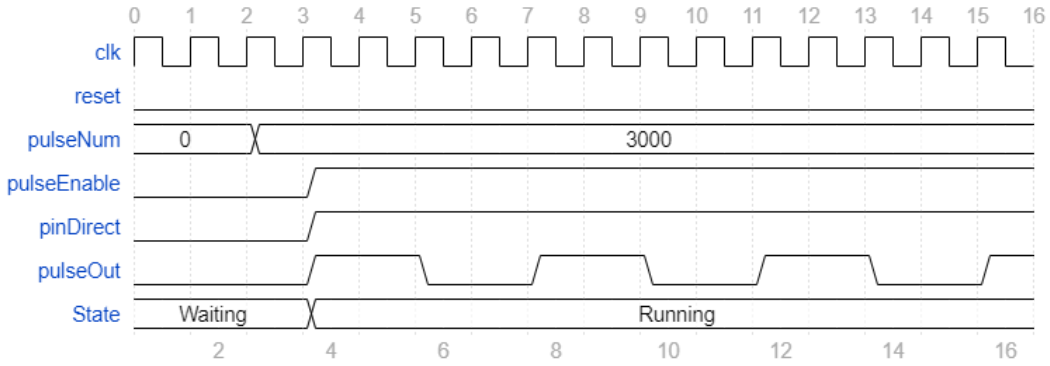


A. Waiting:

The MCC is currently awaiting a valid input for the target position, which can be achieved by following these steps: first, disable the MCC, input the desired target position, and then re-enable the MCC. The figure below illustrates the behavior of the MCC's input and output ports when utilizing the **`reset`** port to disable and subsequently enable the MCC block.

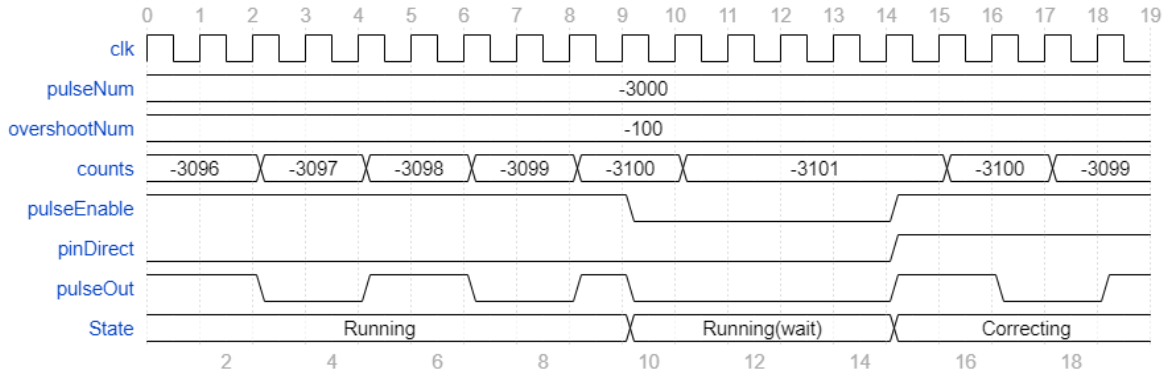


Another way to shift the MCC from the Waiting state to the Running state is by ensuring the MCC remains active while entering a **`pulseNum`** of zero. Afterward, input the intended target position.



B. Running:

The MCC generates a pulse train using **`pulseOut`** and a direction control signal through **`pinDirect`**. While in this mode, the MCC monitors the encoder channel. When the total count of pulses ,**`counts`**, reaches or exceeds the specified sum of **`pulseNum`** and **`overshootNum`**, the MCC suspends the pulse train output by pulling down **`pulseEnable`** and initiates a pause to ensure stabilization of the encoder channel. This waiting interval accommodates the inherent time lag between sending the pulse train to the actuator and receiving the corresponding encoder pulses. Following the conclusion of this waiting period, the MCC moves to the subsequent state, Correcting.

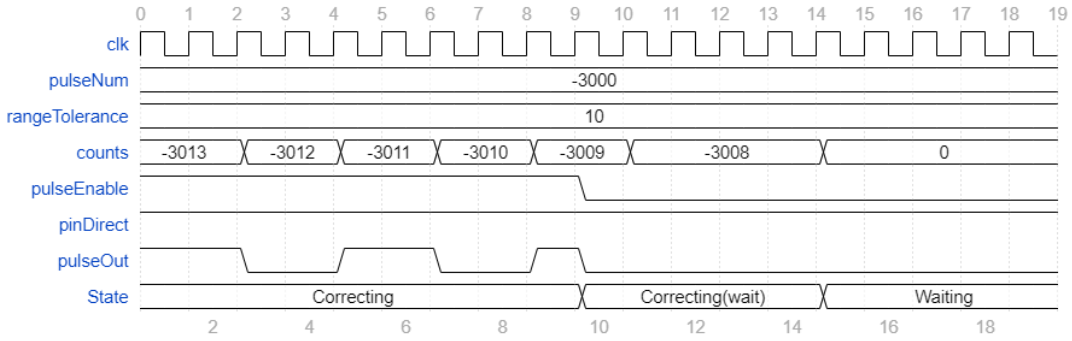


C. Correcting:

Intentionally overshooting the actuator prompts a directional adjustment in this stage, determined by the difference between the target encoder pulse number and the received pulse count. For instance, if the received pulse count exceeds the target encoder pulse number, the **`pinDirect`** is set to the opposite direction to realign the actuator with the target position (**`pulseNum`**). As the actuator approaches the target position (within the specified **`rangeTolerance`**), the **`pulseEnable`** is low, allowing for a waiting period until the encoder channel stabilizes.

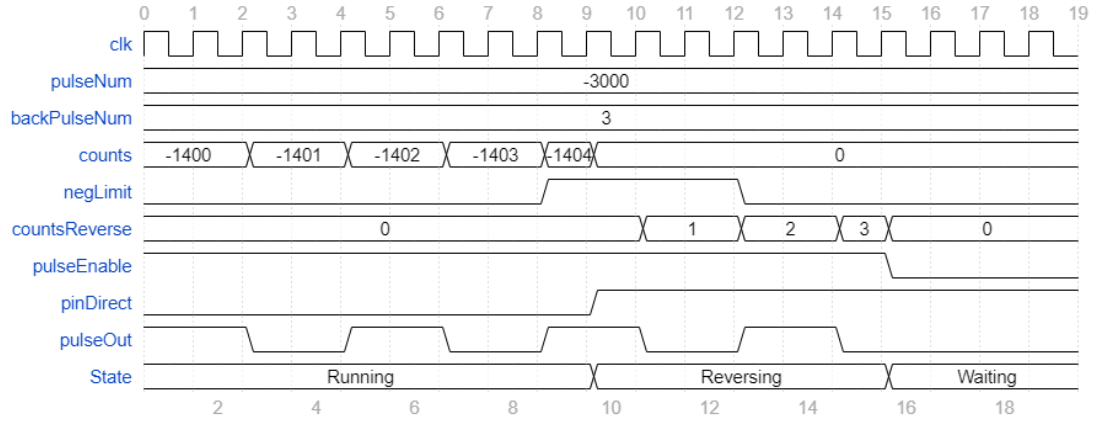
During this pause, if there are pulse count fluctuations beyond the **`rangeTolerance`**, the **`pulseEnable`** is reactivated, initiating another round of corrective actions.

Upon the conclusion of the waiting period and if the encoder pulse counts remain within the tolerance range, the MCC transitions back to the Waiting state.



D. Reversing:

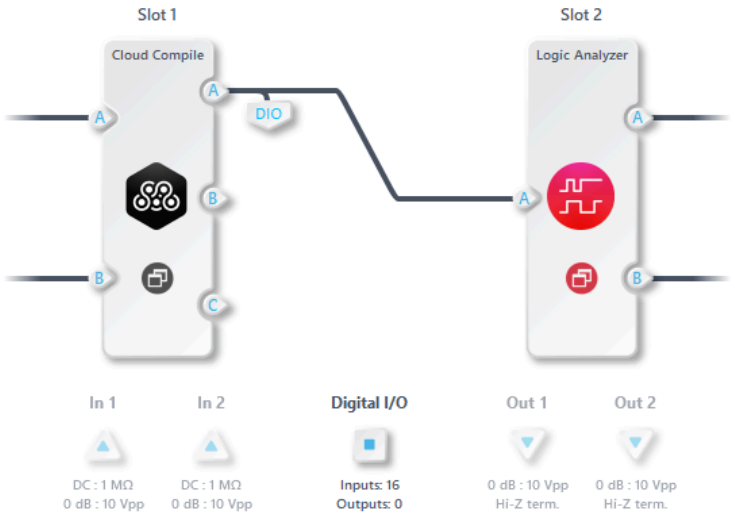
Irrespective of the current state, when any of the limit switches (**`negLimit`** and **`posLimit`**) are activated, the MCC will transition into the Reversing state to maneuver the actuator backward slightly. The specific distance moved backward is controlled by the value specified for **`backPulseNum`**.



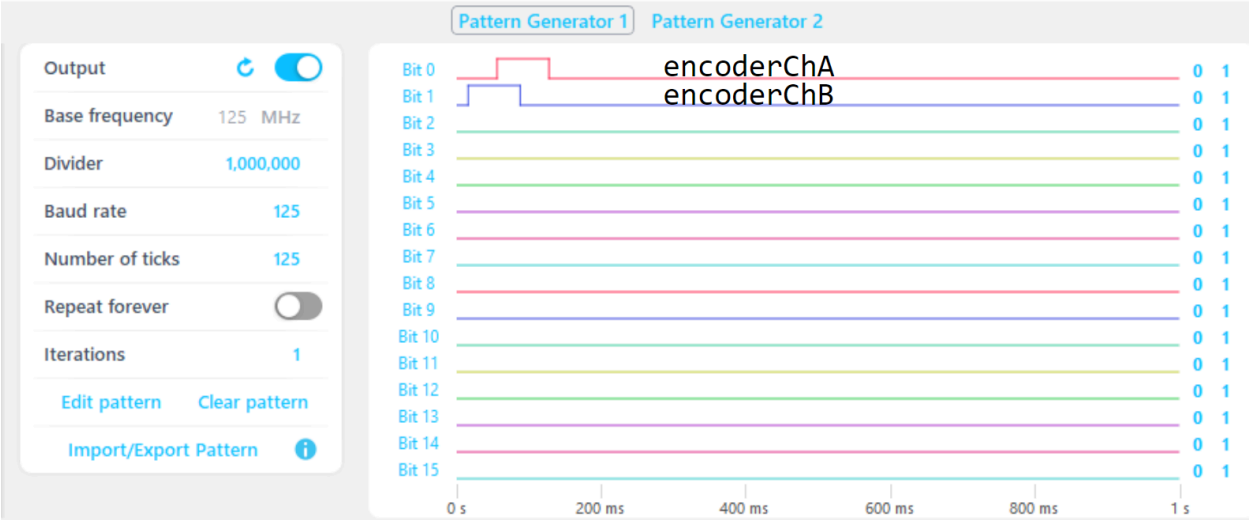
3. Verifications

A. Configurations

Below are the settings for Multi-instrument Mode, MCC registers, and Logic Analyzer/Pattern Generator:



Bits 0 and 1 of Pattern Generator 1 are configured as two pulse train outputs. One channel leads while the other lags, simulating the behavior of the encoder channels. The switch between lagging and leading channels was performed manually according to **`pinDirect`**.



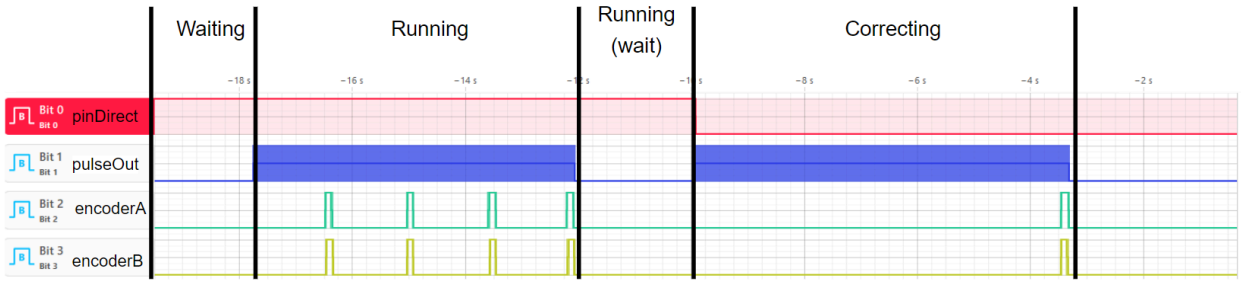
The MCC's control registers are set as follows: the anticipated pulse count **`pulseNum`** from the encoder channels is 4 pulses (equivalent to 8 edges), with an overshoot pulse count **`overshootNum`** of 4 pulses (8 edges). The tolerance range **`rangeTolerance`** is specified as 3 pulses (5 edges). Additionally, the **`waitInterval`** after the MCC reaches the target position is approximately 2.15 seconds (calculated as 67,108,864 * 32 ns).

Register	Decimal (Unsigned)	Decimal (Signed)	Hexadecimal	Binary	
Control0	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000	reset
Control1	262,148	262,148	0004 0004	0000 0000 0000 0100 0000 0000 0000 0100	activeIntervalRun deadIntervalRun
Control2	8,388,736	8,388,736	0080 0080	0000 0000 1000 0000 0000 0000 1000 0000	activeIntervalCorrect deadIntervalCorrect
Control3	8	8	0000 0008	0000 0000 0000 0000 0000 0000 0000 1000	pulseNum
Control4	8	8	0000 0008	0000 0000 0000 0000 0000 0000 0000 1000	overshootNum
Control5	8	8	0000 0008	0000 0000 0000 0000 0000 0000 0000 1000	backPulseNum
Control6	5	5	0000 0005	0000 0000 0000 0000 0000 0000 0000 0101	rangeTolerance
Control7	67,108,864	67,108,864	0400 0000	0000 0100 0000 0000 0000 0000 0000 0000	waitInterval

B. Digital twin tests

While in the Running state, the MCC ceased the pulse train output upon receiving 8 pulses (calculated as **`pulseNum`** + **`overshootNum`**). Subsequently, it paused briefly before transitioning into the Correcting state. Upon entering this state, the **`pinDirect`** signal flipped, signaling a change in the actuator's driving direction.

Upon receiving 2 pulses in the Correcting state, the correction process paused the **`pulseOut`** as the pulse counter fell within the tolerance range. Specifically, 8 pulses (forward) minus 2 pulses (backward) resulted in 6 pulses (forward), which is less than the sum of 4 pulses (**`pulseNum`**) plus the 3-pulse (**`toleranceRange`**) (i.e., 4 + 3 pulses).



The following test is designed to assess the Reversing stage's functionality, whereby the **`pinDirect`** was switched when the **`posLimit`** switch was activated. Following this, the

Reversing stage concluded after receiving 4 pulses, as the predetermined number of reversing pulses **`backPulseNum`** was set to 4 (equivalent to 8 edges).

C. Actuator tests

Using the Time and Frequency Analyser (TFA), the exact number of Encoder signals was able to be counted. A series of tests were conducted, varying the value of **`pulseNum`** over a range of values between 10 and 100,000. The results are documented in the table below.

The TFA is measuring both rising and falling edges for the **`encoderChA`**, so count will be the total number of edges detected for a single channel. If we double this, that will give the approximate number of total edges detected. By subtracting count from **`pulseNum`**, we will obtain the amount of additional edges (which could be from over-travel and/or **`Correcting`**). Since we do not currently have a dedicated instrument to count the net number of edges detected, the closest way to approximate the number of edges traveled is by minimizing the correction pulses. This can be achieved by reducing the duty cycle to minimize overshoot. This however is only practical for reasonably small values of numPulse, as it can take up to 15 seconds to move 10,000 pulses with a low duty cycle.

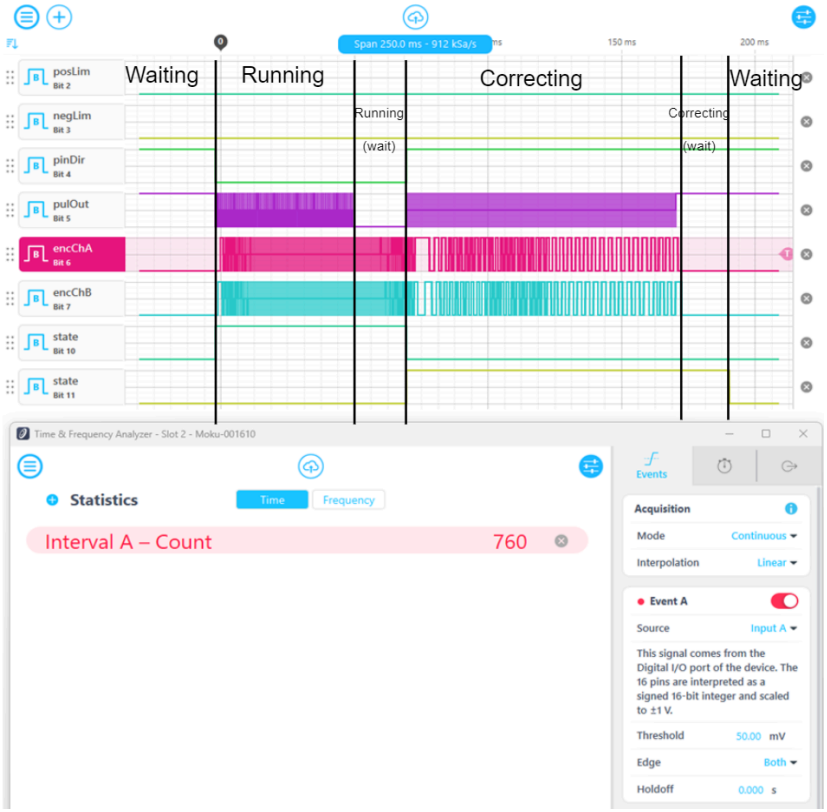
Test no.	pulNum	Count encChA	Approx Combination	Difference	%
1	-10	27	54	-44	440
2	10	27	54	-44	440
3	-100	148	296	-196	196
4	100	90	180	-80	80
5	-500	453	906	-406	81.2
6	500	302	604	-104	20.8
7	-1000	760	1520	-520	52
8	1000	542	1084	-84	8.4
9	-5000	2744	5488	-488	9.76
10	5000	2554	5108	-108	2.16
11	-10000	5228	10456	-456	4.56
12	10000	5222	10444	-444	4.44
13	-20000	10351	20702	-702	3.51
14	20000	10424	20848	-848	4.24
15	-50000	25448	50896	-896	1.792
16	50000	25310	50620	-620	1.24
17	-100000	50774	101548	-1548	1.548
18	100000	50101	100202	-202	0.202

We can see from the data that the actuator tends to travel a distance which is closer to the expected distance when it is traveling forward. This is purely down to the behavior of the experimental setup. The actuator travels slower in one direction than the other when using a duty cycle to control speed (even with a 50% duty cycle). The reduced speed in the forward direction results in the MCC being able to more accurately control the actuator.

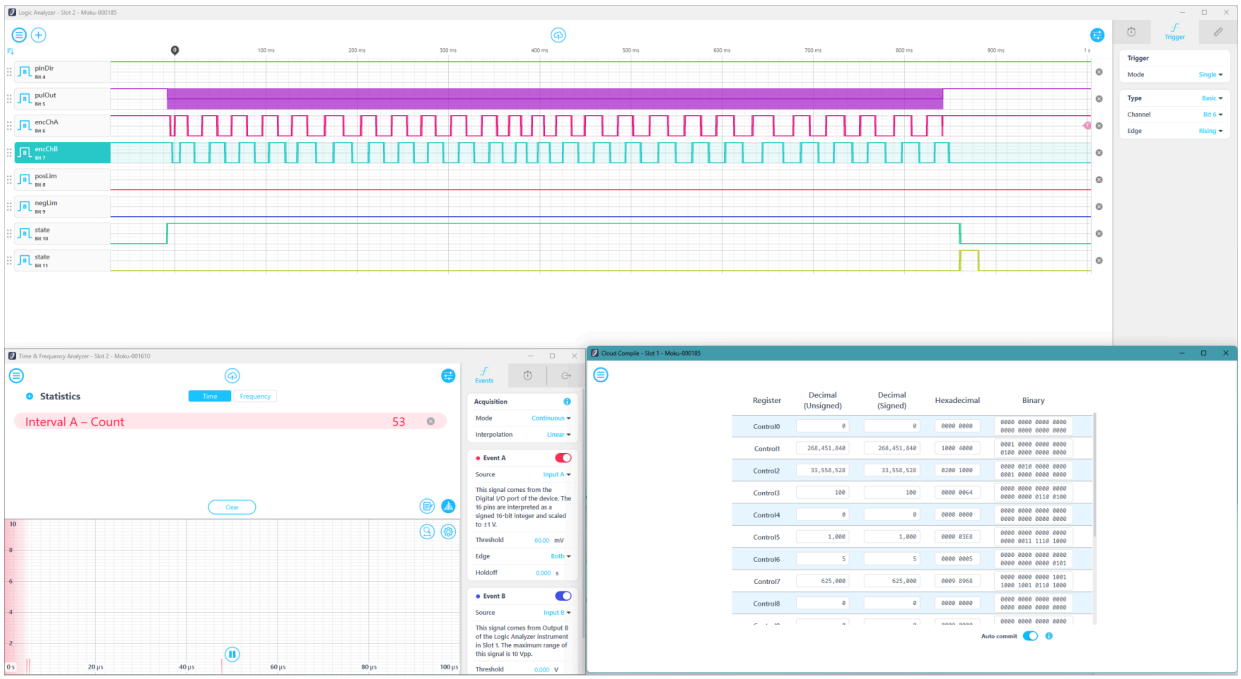
There is also a very large error associated with a small number of pulses, thus a very low duty cycle would be recommended. The value of **`pulseNum`** should be no less than 1000, and a duty cycle of 20% would be typical for steady operation. Note that the tests were carried out with a 33% duty cycle. We observe that for large values of **`pulseNum`**, the overshoot/correction only contributes to a small percentage of the edges detected.

From an operational point of view, we can confirm that the actuator behaves as expected. Tripling the numPulse value makes the actuator travel 3 times further, halving duty cycle makes the actuator move half as quick, and changing the direction of pulseNum changes the direction of travel.

This image shows how the waveforms and count were recorded. The Logic Analyzer clearly displays the **`Running`** and **`Correcting`** stages.



Test case 1: **`pulseNum`** = 100



Test case 2: **`pulseNum`** = 100,000